```r
#Ctrl+Enter for Execute selected code
#Data Type is the type of data
#Variable is a Memory container use for store value

EmpName = "James Smith"
print(EmpName)
typeof(EmpName)

Salary=80000.25
print(Salary)
typeof(Salary)

Empcode=1001L
print(Empcode)
print(typeof(Empcode))

#Data Structure in R Programming
#Vector      =   <-
v1<-c(10,20,30,40,50)
v2<-c(11,22,33,44,55)
v1
v2
v1[2]
v3<-c(60,80,99,70)
v3

c(1:10)
#Matrix store data one data type into rows and column
data<-matrix(c(1:6),nrow=3,byrow=TRUE)
data
data[1][1]

#List store different type of values
#it can be any type like vector, matrix etc
Std_Info<-list("Anita",80.25,"India")
Std_Info

#Array Store Data more then two Dimension store one type data
v1<-c(10,20,30,40)
v2<-c(11,22,33,44)
ary<-array(c(v1,v2),dim=c(2,2,2))
ary

#Data Store Data more then two Dimension store different
#type data   Key,value pair
df = data.frame(
  "ID" = c(1, 2, 3, 4),
  "Name" = c("Tony", "James", "Gita", "Anita"),
  "Age" = c(20, 34, 24, 40)
)
df
df["Name"]
class(df)
length(df)
seq(1,10,2)  #jump by two step

# Arithmetic Operator
vt1<-c(10,11,15)
```

```
vt2<-c(2,3,5)
print(vt1+vt2)
print(vt1-vt2)
print(vt1*vt2)
print(vt1/vt2)    #Show Divide with decimal
print(vt1%/%vt2)  #Divide without decimal
print(vt1%%vt2)   #Modular
print(vt1^vt2)    #Power

#Logical Operator  &&(and), ||(or), !(not)
print(50<60 && 100<500)
print(50<60 || 1000<500)
print(!(50<60))


#relational operator
print(80<90)
print(101>50)
print(70<=70)
print(90>=90)
print(100==150)  #Comparing the values
print(80!=70)

#Assignment operator
n1=100
n2<-120
n3<<-60
n1
n2
n3

80->p1
p1
90->>p2
p2

rg<-2:10  #: colon show range
rg

print(5%in%rg) #in check the existing of values
print(5%*%5)  #it is use for matrix multiplication

#function with argument----------
#Logical
pass<-TRUE
print(pass)
typeof(pass)

#Numeric
age<-25
age
typeof(age)
class(age)

marks<-25.25
marks
class(marks)
```

```
#Integer
num<-505L
num
class(num)
#Complex
cp=3+2i
class(cp)

#Character
grade<-'A'
grade
class(grade)

city<-"Delhi"
city
class(city)

code<-"101"
code
class(code)

#Raw
v<-charToRaw("AB")
v
class(v)

#library
library(pryr)
parenvs(all=TRUE)

#Create New Environment
new.env(hash=TRUE)

newEnv<-new.env()
newEnv$x<-10
print(newEnv$x)

newEnv$value<-1:10
print(newEnv$value)

newEnv$age<-25
ls(newEnv)
ls()

install.packages('plyr')
library(plyr)

x<-c("one","Two")
x[2]
x[[2]]

x1<-1:15
cat("Original Vector",x1)
cat("Original Vector",x1[1:5])
cat("Original Vector",x1[c(1,3,5)])
cat("Original Vector",x1[[1]])

lst<-list(a=10,b=20,c=30,d=40)
```

```
lst
#----[]  [[]]----
cat("First Element ",lst[[1]])

#---$----
cat("First Element ",lst$b)

lst2<-list(11,22,33,44)
lst2
lst2[[3]]

#R Base Package
airq<-subset(airquality,Temp<65,select = c(Month))
print(airq)
View(airquality)

#create dataset for reshape the data
data1 <- data.frame(
  "idvar"= c(1,2,3),
  "Income2008" = c(100,87,94),
  "Income2009" = c(72,65,46)
)
data1



#--------------------------------
data <- data.frame(group_1 = rep(c("A", "B", "C"), each = 3),
                   group_2 = LETTERS[4:6],
                   values = 1:9)
data

#reshape data
data_reshape1 <- reshape(data,                                # Applying
reshape function
                         idvar = "group_1",
                         timevar = "group_2",
                         direction = "wide")
data_reshape1

#----------------------------
install.packages("tidyr")          # Install & load tidyr package
library("tidyr")
data2 <- spread(data,key = group_2,value = values)
data2

#-------------------------------
#install.packages("dplyr")
library(dplyr)
# Create a data frame
d2 <- data.frame( name = c("Abhi", "Bhavesh", "Arun", "Dimri"),
                  age = c(17, 7, 9, 16) )
d2
# Arranging name according to the age
d2<- arrange(d2, age)
print(d2)
```

```
# R program to reorder a data set

# Loading library
library(dplyr)

# Calling dataset
x <- BOD
x
# Calling arrange() function
arrange(x, demand)

#----------------------------
#Data importing in CSV Files
# getwd() #show current working directory
# # # [1] "C:/Users/Sachin sirohi/Documents"
setwd("F:\\Software\\Data Science\\R Programming Notes\\dataset")
# for comment ctrl+shift+c

# Import the data using read.csv()
myData = read.csv("CardioGoodFitness.csv",stringsAsFactors=F)
# Print the first 6 rows
print(head(myData))

# Compute the mean value
mean = mean(myData$Age)
print(mean)

# Compute the median value
median = median(myData$Age)
print(median)

#-------mode-------------------
library(modeest)
# Compute the mode value
mode = mfv(myData$Age)
print(mode)

#-------
# Calculate the maximum
max = max(myData$Age)
# Calculate the minimum
min = min(myData$Age)
# Calculate the range
range = max - min

cat("Range is:\n")
print(range)

# Alternate method to get min and max
r = range(myData$Age)
print(r)

#------------------------
# Calculating variance
variance = var(myData$Age)
print(variance)
```

```
#--------------------
# Calculating Standard deviation
std = sd(myData$Age)
print(std)

#-----------------------
# Calculating Quartiles
quartiles = quantile(myData$Age)
print(quartiles)

#--------------------
# Calculating IQR
IQR = IQR(myData$Age)
print(IQR)

#-----------------
# Calculating summary
summary = summary(myData$Age)
print(summary)

#---------------------
#Calculating summary
summary = summary(myData)
print(summary)

#loop iterate the set of instruction untill the condition hold true
#apply function
data=matrix(c<-(1:50),nrow=5,ncol=6)
data
m1<-apply(data,2,sum)
m1

#lapply function
color<-c("RED","GREEN","BLUE")
color
color<-lapply(color,tolower)
paste(color,sep=" ")

#sapply function
dt<-cars
amin=lapply(dt,min)
smin=sapply(dt,min)
paste(amin,sep=" ")
paste(smin,sep=" ")

#tapply
data=read.csv('iris.csv')
print(head(data))
tm=tapply(data$sepal_width,data$species,mean)
tm

#rep function
r<-rep(c("A", "B", "C"), each = 2)
r

z<-c(10,20,30)
r2<-rep(z,c(3,1,2)) #repeat value 3 1 2
r2
```

```
#Mapply
m=mapply(rep,1:4,5)
m

#Condition use for checking
marks=10
if(marks>=40)
{
  print("Pass")
}else
{
  print("Fail")
}

#Multiple condition
age=50
if(age>=18 && age<=60)
{
  print("Eligible for Drive")
}else if(age>1 && age<18)
{
  print("Not Eligible for Drive ")
}else if(age>60 && age<=100)
{
  print("Senior Citizen ")
}else
{
  print("Invalid Information ")
}

#-------While loop execute untill the condition hold true
a<-1
while(a<=5)
{
  cat("While Loop ",a,"\n")
  a<-a+1
}

#use condition inside the loop
b<-1   #initialization
while(b<=15)  #condition
{
  if(b<=7)
  {
    cat("While Loop with codition ",b,"\n")
  }
  b<-b+1    #increment
}

#For Loop
for(x in 1:5)
{
  print("for loop")
}

#break use for exit and next use for skip
for(x in 1:5)
```

```
{
  if(x==3)
  {
    break
  }
  print("for loop with break")
}
#next
for(x in 1:5)
{
  if(x==3)
  {
    next
  }
  cat("for loop with next",x,"\n")
}

#-----------Aggregate function----------
df<-data.frame(team=c('A','A','A','B','B','B'),
               position=c('G','G','F','G','F','F'),
               points=c(99,90,88,95,99,80),
               assists=c(33,28,31,39,34,23),
               rebounds=c(30,28,24,24,28,33))
df

aggregate(df$points,by=list(df$team),FUN=mean)
aggregate(df$points,by=list(df$team),FUN=length)
aggregate(df$points,by=list(df$team),FUN=sum)
aggregate(df$points,by=list(df$team,df$position),FUN=mean)

#Function Table
install.packages('psych')
library(psych)
describe(df)
describeBy(df,group=df$position)

df<-data.frame(team=c('A','A','B','B','C','C','C'),
               points=c(15,22,29,41,30,11,19),
               rebounds=c(7,8,6,6,7,9,13),
               steals=c(1,1,2,3,5,7,5)
               )
df
describe(df)
describe(df[,c('points','rebounds')],fast=TRUE)

# Type casting
val='10'
typeof(val)
print(val+50)
val2=as.integer(val)
typeof(val2)
print(val2+10)

#--------Data Manipulation-------------
install.packages('dplyr')
library('dplyr')
library('datasets')
data(iris)
```

```
summary(iris)

#select()
selected<-select(iris,Sepal.Length,Sepal.Width,Petal.Length)
head(selected)

selected<-select(iris,Sepal.Length,Petal.Length)
head(selected,3)

selected<-select(iris,c(1:3))
head(selected,2)

#hide (-)minus use for hide particular column
selected<-select(iris,-Sepal.Length,Petal.Length)
selected

#filter()
filt=filter(iris,Species=="setosa")
head(filt,3)


#filter()
filt2=filter(iris,Species=="versicolor",Sepal.Width>3)
tail(filt2,3)


#mutate create a new column into exiting dataset
col1<-mutate(iris,Greater.Half=Sepal.Width>0.5*Sepal.Length)
tail(col1)
col1
table(col1$Greater.Half)

#Arrange
#by default ascending
arg<-arrange(col1,Sepal.Width)
head(arg)
#descending
arg2<-arrange(col1,desc(Sepal.Width))
head(arg2)

#summarize use to find mean, mode etc.
sumz<-summarise(arg2,mean.width=mean(Sepal.Width))
head(sumz)

#Grouping
gp<-group_by(iris,Species)
mn<-summarise(gp,mean.sepal=mean(Sepal.Width))
head(mn)

mn<-summarise(gp,mean.sepal=sum(Sepal.Width))
head(mn)

#Pipe Operator
iris%>%filter(Species=="setosa",Sepal.Width>3.8)

#----------Module-8-----------Graph for data visualization-----------
data() #check all datasets
View(Orange) #show particular dataframe
```

```r
setwd("F:\\Software\\Data Science\\R Programming Notes\\dataset")
orange<-Orange[,c("age","circumference")]
png(file="OrangeScatterImage.png")
plot(x=orange$age,y=orange$circumference,xlab="Age",
     ylab="circumference",main="Age Vs Circumference",
     col.lab="red",col.main="blue",
     col.axis="darkgreen")
#use for save the file
dev.off()

#------iris dataset----------
# View(iris)
plot(x=iris$Sepal.Length,y=iris$Sepal.Width,xlab="Sepal Length",
     ylab="Sepal Width",main="Iris Dataset",
     col.lab="red",col.main="blue",
     col.axis="purple")

#--------boxplot
x<-c(42,21,24,25,30,29,22,23,24,
     28,32,45,39,40)
boxplot(x,xlab="Box Plot",ylab="Age",
        col.axis="red",col.lab="darkgreen",
        col=c("green"),notch = TRUE)

#---------------
computer<-c(50,90,10,60,78,35,64)
math<-c(70,32,68,97,41,56,1,95)
# boxplot(iris$Sepal.Length~iris$Sepal.Width)
boxplot(computer,math,names=c("S_Computer","S_Math"),
        col=c("green","yellow"))

#-------Barplot chart
Marks<-c(45,87,40,90,50)
Sub<-c("Hindi","English","Math","Science","Computer")
barplot(Marks,names=Sub,xlab = "Subject",
        ylab = "Marks",col="orange",
        main="Student Information",border="blue")

#----------Stacked Bar
Sub<-c("Hindi","English","Math","Science","Computer")
colors=c("red","green","blue")
cls<-c("B.A","B.Com")
values<-matrix(c(45,87,40,90,50,60),nrow=2,ncol=3,byrow=TRUE)
barplot(values,names=Sub,xlab = "Subject",
        ylab = "Marks",col=colors,
        main="Student Information",border="blue")
legend("topright",cls,cex = 1.3,fill=colors)

#-------histogram Chart
x1<-c(50,90,10,60,78,35,64,80,90,
     70,45,65,63,78,20,35,21,70,45)
hist(x1,xlab = "Bins",ylab = "Values",main="Histogram Chart-1",
     col="green",border = "red")

#breaks use for binns
x2<-c(50,90,10,60,78,35,64,80,90,
      70,45,65,63,78,20,35,21,70,45)
```

```
hist(x2,breaks=5,xlab = "Bins",ylab = "Values",main="Histogram Chart-2",
     col="yellow",border = "red")

#------Pie Chart
x<-c(60,80,40,90)
c_name<-c("London","New York","Singpore","India")
pie(x,labels=c_name,col=rainbow(length(x)))

#-----Pie Chart using more feature
#cex use for far away the legend from chart
x<-c(60,80,40,90)
percentage<-round(100*x/sum(x),2)
c_name<-c("London","New York","Singpore","India")
pie(x,labels=percentage,col=rainbow(length(x)),main="Country Info")
legend("topleft",c_name,fill=rainbow(length(x)),cex=0.7)


#------pie chart
install.packages('plotrix')
library('plotrix')
x<-c(60,80,40,90)
c_name<-c("London","New York","Singpore","India")
pie3D(x,labels=c_name,col=rainbow(length(x)),
      main="Country Info",explode = 0.1)


#----------------Module-9 Advance
Chart------------------------------------------
install.packages('ggplot2')
library('ggplot2')

df<-data.frame(dose=c("D0.5","D1","D2"),len=c(1.5,8,25))
df
#Scatter Chart
ggplot(data=df,aes(x=dose,y=len))+geom_point()

#Scatter chart using line
ggplot(data=df,aes(x=dose,y=len,group=1))+geom_line()+geom_point()

#Scatter chart using smooth line
df<-data.frame(dose=c("D0.5","D1","D2"),len=c(1.5,8,25))
df
ggplot(data=df,aes(x=dose,y=len,group=1))+geom_point()+geom_smooth(method
="lm")

#Scatterplot using midwest dataset
data()
library('ggplot2')
ggplot(midwest,aes(x=area,y=poptotal))+geom_point()+geom_smooth(method
="lm")

#delete outside range of data
library('ggplot2')
g<-ggplot(midwest,aes(x=area,y=poptotal))+geom_point()+geom_smooth(method
="lm")
g+xlim(c(0,0.1))+ylim(c(0,1000000))
```

```
#------Zoom-------
g<-ggplot(midwest,aes(x=area,y=poptotal))+geom_point()+geom_smooth(method
="lm")
g1<-g+coord_cartesian(xlim=c(0,0.1),ylim=c(0,1000000))
g1
#-------------------------
g<-ggplot(midwest,aes(x=area,y=poptotal))+geom_point()+geom_smooth(method
="lm")
g1<-g+coord_cartesian(xlim=c(0,0.1),ylim=c(0,1000000))

#Add Title and Labels
g1+labs(title = "Area Vs Population",subtitle="From midwest dataset",
        y="Population", x="Area",caption = "Midwest Demographics")
#or
g1+ggtitle("Area Vs Population",subtitle = "From midwest dataset")+
  xlab("Area")+ylab("Population")

#----------------------------
ggplot(midwest,aes(x=area,y=poptotal))+geom_point(col="steelblue",size=3)+
  geom_smooth(method ="lm",col="firebrick")+
  coord_cartesian(xlim=c(0,0.1),ylim=c(0,1000000))+
labs(title="Area Vs Population",subtitle="From midwest dataset",
       y="Population",x="Area",caption="Midwest Demographics")


#---------------------------------
ggplot(midwest,aes(x=area,y=poptotal))+
  geom_point(aes(col=state),size=4)+
  geom_smooth(method ="lm",col="firebrick",size=3)+
  coord_cartesian(xlim=c(0,0.1),ylim=c(0,1000000))+
  labs(title="Area Vs Population",subtitle="From midwest dataset",
  y="Population",x="Area",caption="Midwest Demographics")
plot(gg)

#--------------------------------

#gg+scale_color_brewer(palette = "Set1")

#--Ggplot Barplot--------------------------------
survey <- data.frame(fruit=c("Apple", "Banana", "Grapes", "Kiwi", "Orange",
"Pears"),
                     people=c(40, 50, 30, 15, 35, 20))
survey
#------------------------
library(ggplot2)
ggplot(survey, aes(x=fruit, y=people)) +
  geom_bar(stat="identity")

#-----coloring bar------------------
ggplot(survey, aes(x=fruit, y=people, fill=fruit)) +
  geom_bar(stat="identity")

#-----colors manually-------------
ggplot(survey, aes(x=fruit, y=people, fill=fruit)) +
  geom_bar(stat="identity") +
  scale_fill_manual(values=c("red2", "yellow2", "slateblue4", "green3",
"orange", "olivedrab2"))
```

```
#-------preset color schemes using scale_fill_brewer()
ggplot(survey, aes(x=fruit, y=people, fill=fruit)) +
  geom_bar(stat="identity") +scale_fill_brewer(palette="OrRd")


#-------changing plot background theme--------------------
# Change the ggplot theme to 'Minimal'
ggplot(survey, aes(x=fruit, y=people, fill=fruit)) +
  geom_bar(stat="identity") +
  theme_minimal()
#some other themes
# theme_classic()
# theme_bw()
# theme_dark()
# theme_gray()
# theme_linedraw()
# theme_light()

#----------Ggplot Heat Map-------
#-------Example Data
#Set seed for reproducibility
set.seed(123)
data <- matrix(rnorm(100, 0, 10), nrow = 10, ncol = 10)
colnames(data) <- paste0("col", 1:10)
rownames(data) <- paste0("row", 1:10)
print(data)

#-----------heapmap applied---
heatmap(data)

#----------------------
#reorder data with the help of melt

install.packages("reshape")

library("reshape")
data_melt <- melt(data)
head(data_melt)

#-------------------------------
install.packages("ggplot2")
library("ggplot2")

ggp <- ggplot(data_melt, aes(X1, X2)) +
  geom_tile(aes(fill = value))
ggp

#--------------------------------
ggp + scale_fill_gradient(low = "green", high = "black")


#------------------Module-10---Linear
Regression----------------------------
#Linear Model
View(women)
#-------------------------
# attach(women)
# library(psych)
```

```
# describe(women) [, c(1:5,8:9)]   # selected columns
summary(women)
#------------------------
plot(women$height,women$weight)

#----------------------------
fit <- lm(weight ~ height, data = women)
fit
#---------------------------
summary(fit)

#--------------------
data()
dataset=read.csv('Salary.csv')
dataset
#--------------------
install.packages('caTools')
library(caTools)

#----Splitting the dataset----------------------
# set.seed(123)
#splitting the dataset into the
#training set and Test set

split=sample.split(dataset$Salary, SplitRatio= 0.7)
trainingset=subset(dataset, split==TRUE)
testset=subset(dataset, split==FALSE)

#-------linear regression------
lm.r= lm(formula=Salary ~YearsExperience, data=trainingset)
coef(lm.r)
#Predicting the test set resuslts
y_pred=predict(lm.r, newdata=testset)
y_pred
#------visiualization of the training set results-------------------------
library(ggplot2)
ggplot()+
  geom_point(aes(x=trainingset$YearsExperience, y=trainingset$Salary),
colour='pink',size=5)+
  geom_line(aes(x=trainingset$YearsExperience, y=predict(lm.r,
newdata=trainingset)), colour='green') +
  ggtitle('Salary vs Experience(Training set)')+
  xlab('Years of Experience')+
  ylab('Salary')

#----visiualization of the test set results--------------------------
ggplot()+
  geom_point(aes(x=testset$YearsExperience, y=testset$Salary),
colour='yellow',size=5)+
  geom_line(aes(x=trainingset$YearsExperience, y=predict(lm.r,
newdata=trainingset)), colour='blue') +
  ggtitle('Salary vs Experience(Test set)')+
  xlab('Years of Experience')+
  ylab('Salary')

#---------------
coef(lm.r)
```

```
#------Multiple linear Regression------------
data()

#----------
# Importing the dataset
dataset = read.csv('data2.csv')
dataset
# Encoding categorical data
dataset$State = factor(dataset$State,
                       levels = c('New York', 'California', 'Florida'),
                       labels = c(1, 2, 3))
dataset$State

#------# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123) #123 is set as the random number value.
split = sample.split(dataset$Profit, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)

# Fitting Multiple Linear Regression to the Training set
regressor = lm(formula = Profit ~ .,
               data = training_set)

# Predicting the Test set results
Y_prediction = predict(regressor, newdata = test_set)
regressor
#------
Y_prediction






# start from 10-11-2023
#---------------Start from next class----------------------------

#--------------Module-11--Logistic Regression---------------------

# Generalized Linear Models
# Generalized linear models are fit using the glm( ) function. The form of
the glm function is

# glm(formula,data,family)

# Select some columns form mtcars.
input <- mtcars[,c("am","cyl","hp","wt")]
print(head(input))


#---------------------------------
```

```
# Create Regression Model
# We use the glm() function to create the regression model and get its
summary for analysis.

input <- mtcars[,c("am","cyl","hp","wt")]
am.data = glm(formula = am ~ cyl + hp + wt, data = input, family =
binomial)
print(summary(am.data))

#----------Predict()----------------------------
# predict(model,newdata,type)
# model A regression model, usually the result of lm() or glm().
# newdata A data.frame giving the values of the predictor(s) to use in the
prediction of the response variable.
# type The type of prediction, usually you want type = "response".

nd = data.frame(macro = c(40, 50, 60))
nd
#-------------
head(mtcars)
#----------------
#fit logistic regression model
model <- glm(am ~ disp + hp, data=mtcars, family=binomial)
model

#view model summary
summary(model)

#-------------------------------
#define new data frame
newdata = data.frame(disp=c(200, 180, 160, 140, 120, 120, 100, 160),
                     hp=c(100, 90, 108, 90, 80, 90, 80, 90),
                     am=c(0, 0, 0, 1, 0, 1, 1, 1))

#view data frame
newdata

#use model to predict value of am for all new cars
newdata$am_prob <- predict(model, newdata, type="response")

#view updated data frame
newdata

#-------------------------------
#create vector that contains 0 or 1 depending on predicted value of am
am_pred = rep(0, dim(newdata)[1])
am_pred[newdata$am_prob > .5] = 1

#create confusion matrix
table(am_pred, newdata$am)

#calculate percentage of observations the model correctly predicted
response value for
mean(am_pred == newdata$am)
# [1] 0.75

#for more details
```

```
#https://www.dataanalytics.org.uk/logistic-regression-model-prediction/
#predict


#---------------Module-12--Decision Tree Classification-------------
install.packages("rpart")
library(rpart)

# Create decision tree using regression
fit <- rpart(Sepal.Width ~ Sepal.Length +
               Petal.Length + Petal.Width + Species,
             method = "anova", data = iris)
#------------------------------
#Plot the tree

plot(fit, uniform = TRUE,
     main = "Sepal Width Decision
                 Tree using Regression")
text(fit, use.n = TRUE, cex = .7)

#-------------------------------
# Print the decision tree model
# Print model
print(fit)

#---------------------------------
# Predicting the sepal width

# Create test data
df  <- data.frame (Species = 'versicolor',
                   Sepal.Length = 5.1,
                   Petal.Length = 4.5,
                   Petal.Width = 1.4)

# Predicting sepal width
# using testing data and model
# method anova is used for regression
cat("Predicted value:\n")
predict(fit, df, method = "anova")

#--------------------------------
# Classification Tree
# Load Library
install.packages('DAAG')
install.packages('party')
install.packages('rpart')
install.packages('rpart.plot')
install.packages('mlbench')
install.packages('caret')
install.packages('pROC')
install.packages('tree')
library(DAAG)
library(party)
library(rpart)
library(rpart.plot)
library(mlbench)
library(caret)
library(pROC)
```

```
library(tree)

#Getting Data -Email Spam Detection
str(spam7)

#--------------------
mydata <- spam7
mydata

#--------------------
set.seed(1234)
ind <- sample(2, nrow(mydata), replace = T, prob = c(0.5, 0.5))
train <- mydata[ind == 1,]
test <- mydata[ind == 2,]
# Tree Classification
tree <- rpart(yesno ~., data = train)
rpart.plot(tree)

#------------------------
printcp(tree)

#------------------------
rpart(formula = yesno ~ ., data = train)

#------------------------
plotcp(tree)

#------------------------
tree <- rpart(yesno ~., data = train,cp=0.07444)


#-----Model Building-------------------------
# ctree(formula,data)

install.packages("party")
library(party)
print(head(readingSkills))

#---------------------------------
# Load the party package. It will automatically load other
# dependent packages.
library(party)

# Create the input data frame.
input.dat <- readingSkills[c(1:105),]

# Create the tree.
output.tree <- ctree(
  nativeSpeaker ~ age + shoeSize + score,
  data = input.dat)

# Plot the tree.
plot(output.tree)


#-------Pruning The Tree-----------------------------
#pre-pruning
install.packages("rpart")
```

```
library(rpart)
hr_data <- read.csv("hr.csv")
hr_data
#---------------------------------
sample_ind <- sample(nrow(hr_data),nrow(hr_data)*0.70)
train <- hr_data[sample_ind,]
test <- hr_data[-sample_ind,]
#----------------------------------
#Base Model
hr_base_model <- rpart(left ~ ., data = train, method = "class",
                       control = rpart.control(cp = 0))
summary(hr_base_model)
#Plot Decision Tree
plot(hr_base_model)
# Examine the complexity plot
printcp(hr_base_model)
plotcp(hr_base_model)
#----------------------------
printcp(hr_base_model)
#----------------------------

# Compute the accuracy of the pruned tree
hr_model_preprun <- rpart(left ~ ., data = train, method = "class",
                          control = rpart.control(cp = 0, maxdepth =
8,minsplit = 100))
# Compute the accuracy of the pruned tree
test$pred <- predict(hr_model_preprun, test, type = "class")
accuracy_preprun <- mean(test$pred == test$left)

#------Post pruning----------------------------
#Postpruning
# Prune the hr_base_model based on the optimal cp value
hr_model_pruned <- prune(hr_base_model, cp = 0.0084 )
# Compute the accuracy of the pruned tree
test$pred <- predict(hr_model_pruned, test, type = "class")
accuracy_postprun <- mean(test$pred == test$left)
data.frame(base_accuracy, accuracy_preprun, accuracy_postprun)
```

```
#-----Module -13 Clustring Analysis---------------------------------------
#hierarchical Clustering
install.packages("dplyr")
library(dplyr)
head(mtcars)

#-------------------------------
# Finding distance matrix
distance_mat <- dist(mtcars, method = 'euclidean')
distance_mat
```

```
# Fitting Hierarchical clustering Model
# to training dataset
set.seed(240)  # Setting seed
Hierar_cl <- hclust(distance_mat, method = "average")
Hierar_cl

# Plotting dendrogram
plot(Hierar_cl)

# Choosing no. of clusters
# Cutting tree by height
abline(h = 110, col = "green")

# Cutting tree by no. of clusters
fit <- cutree(Hierar_cl, k = 3 )
fit

table(fit)
rect.hclust(Hierar_cl, k = 3, border = "blue")

#-----------------------------
#The values are shown as per the distance matrix calculation with the
method as euclidean
Hierar_cl

#----K-Means Clustring------------------------
# Loading data
data(iris)

# Structure
str(iris)
#---------------------
#-----------------------------------
# Performing K-Means Clustering on Dataset
# Using K-Means Clustering algorithm on the dataset which includes 11
persons and 6 variables or attributes
# Installing Packages
install.packages("ClusterR")
install.packages("cluster")

# Loading package
library(ClusterR)
library(cluster)

# Removing initial label of
# Species from original dataset
iris_1 <- iris[, -5]
iris_1

# Fitting K-Means clustering Model
# to training dataset
set.seed(240) # Setting seed
kmeans.re <- kmeans(iris_1, centers = 3, nstart = 20)
kmeans.re

# Cluster identification for
# each observation
```

```
kmeans.re$cluster

# Confusion Matrix
cm <- table(iris$Species, kmeans.re$cluster)
cm

# Model Evaluation and visualization
plot(iris_1[c("Sepal.Length", "Sepal.Width")],col="green")
plot(iris_1[c("Sepal.Length", "Sepal.Width")],
     col = kmeans.re$cluster)
plot(iris_1[c("Sepal.Length", "Sepal.Width")],
     col = kmeans.re$cluster,
     main = "K-means with 3 clusters")

## Plotiing cluster centers
kmeans.re$centers
kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")]

# cex is font size, pch is symbol
points(kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")],
       col = 1:3, pch = 8, cex = 3)

## Visualizing clusters
y_kmeans <- kmeans.re$cluster
clusplot(iris_1[, c("Sepal.Length", "Sepal.Width")],
         y_kmeans,
         lines = 0,
         shade = TRUE,
         color = TRUE,
         labels = 2,
         plotchar = FALSE,
         span = TRUE,
         main = paste("Cluster iris"),
         xlab = 'Sepal.Length',
         ylab = 'Sepal.Width')

#--------------------------
cm

#----Euclidean Distance-------------------------
data("USArrests")
df <- USArrests
#--------------------
#Remove any missing value that may be present in the data
df <- na.omit(df)

#Scale of Data
df <- scale(df)
head(df,n=3)

#---------------------
#clustering Distance Measures.
set.seed(123)
ss <- sample(1:50,15) # get the row index of randomly selected 15 rows
df <- USArrests[ss,] # subset the rows basis this row index
df.scaled <- scale(df)

#----------------------------
```

```
dist.eucl <- dist(df.scaled,method="euclidean")

#----------------------------------
# subset the first three columns and rows and round the values
round(as.matrix(dist.eucl)[1:3,1:3],1)

#--------------------------------------
manhattanDistance <- function(vect1, vect2){
  dist <- abs(vect1 - vect2)
  dist <- sum(dist)
  return(dist)
}

# Initializing a vector
vect1 <- c(3, 6, 8, 9)

# Initializing another vector
vect2 <- c(1, 7, 8, 10)


print("Manhattan distance between vect1 and vect2 is: ")

# Call the function to calculate Manhattan
# distance between vectors
manhattanDistance(vect1, vect2)


#-------------Module-14 --Calculating Distance---------------------

# Loading data
dataset = read.csv('Market_Basket_Optimisation.csv')
# Structure
dataset

#------------------------------
dataset = read.transactions('Market_Basket_Optimisation.csv',
                            sep = ', ', rm.duplicates = TRUE)

# Structure
str(dataset)

#-------------------------
# Performing Association Rule Mining on Dataset
# Installing Packages
install.packages("arules")
install.packages("arulesViz")

# Loading package
library(arules)
library(arulesViz)

# Fitting model
# Training Apriori on the dataset
set.seed = 220 # Setting seed
associa_rules = apriori(data = dataset,
                        parameter = list(support = 0.004,
                                         confidence = 0.2))
```

```
# Plot
itemFrequencyPlot(dataset, topN = 10)

# Visualising the results
inspect(sort(associa_rules, by = 'lift')[1:10])
plot(associa_rules, method = "graph",
     measure = "confidence", shading = "lift")




#--------Module-15--Date Manipulation------------------------
dv<-as.Date("2012-05-28")
print(dv)

dv1<-as.Date("01/22/2015",format='%m/%d/%y')
print(dv1)

dv2<-as.Date("15 April, 2020",format='%d %B, %y')
print(dv2)

dv22<-as.Date("15 May, 2020",format='%d %B, %y')
print(dv22)

Sys.Date()
Sys.time()

Sys.timezone()

Sys.time()

#install.packages("lubridate")
library(lubridate)
now()

x<-c("2020-01-01","2018-04-01","2005-07-01")
print(x)

year(x)
month(x)
month(x,label = TRUE)
mday(x)
wday(x,label=TRUE,abbr=FALSE)

print(x)
year(x)
month(x)
month(x,label = TRUE,abbr=FALSE)


x<-c("2020-01-01","2018-04-01","2005-07-01")
print(x)

x<-ymd(x)
print(x)

x+years(1)
x+months(1)
```

```
mday(x)<-c(12,18,24)
print(x)
new_x<-update(x,year=c(2005,2013,1993),month=c(4,6,8),day=c(10,12,25))
print(new_x)
```

```
#--------Module-16--Time Series Analysis------
# save a numeric vector containing 72 monthly observations
# from Jan 2009 to Dec 2014 as a time series object
myvector <- c(10, 9, 8, 7, 2)
myts <- ts(myvector, start=c(2009, 1), end=c(2014, 12), frequency=12)
myts
# subset the time series (June 2014 to December 2014)
myts2 <- window(myts, start=c(2014, 6), end=c(2014, 12))
myts2
# plot series
plot(myts)

#-------------------
# Seasonal decomposition
fit <- stl(myts, s.window="period")
plot(fit)
#------------------
# additional plots
monthplot(myts)
#install.packages('forecast')
library(forecast)
seasonplot(myts)
#-------------------------------------
# simple exponential - models level
fit <- HoltWinters(myts, beta=FALSE, gamma=FALSE)
# double exponential - models level and trend
fit <- HoltWinters(myts, gamma=FALSE)
# triple exponential - models level, trend, and seasonal components
fit <- HoltWinters(myts)

# predictive accuracy
library(forecast)
accuracy(fit)

# predict next three future values
library(forecast)
forecast(fit, 3)
plot(forecast(fit, 3))

#------------------------------------------
# fit an ARIMA model of order P, D, Q

fit <- arima(myts, order=c(p, d, q))

# predictive accuracy
library(forecast)
```

```
accuracy(fit)

# predict next 5 observations
library(forecast)
forecast(fit, 5)
plot(forecast(fit, 5))

#----------------------------------------
library(forecast)
# Automated forecasting using an exponential model
fit <- ets(myts)

# Automated forecasting using an ARIMA model
fit <- auto.arima(myts)

#-------zoo package--------------------------------
#install.packages("zoo")
require("zoo")
# read.table(Filepath,header=TRUE,sep=",",stringsAsFactors=FALSE)->inData


sales<-c(300,400,100,400,800)

sales.data<-ts(data=sales,start=c(2018,1),frequency = 1)
sales.data
#---------------
zoo(sales.data)
#--------------
days<-seq(as.Date("2018-01-01"),as.Date("2018-01-05"),by="days")
zoo(sales,days)

#----The xts library-------------
library(xts)
days <- seq(as.Date("2018-01-01"), as.Date("2018-01-05"), by = "days")

xts(sales, days)

#-----Module-17-Resource Git Library----------
```